Shadow Documentation

Release 0.3.2

Bobby

Oct 18, 2019

Contents:

CHAPTER 1

Shadow

A comprehensive command line utility to render templates and ease code generation.

- Free software: GNU General Public License v3
- Documentation: https://shadow.readthedocs.io.

1.1 Features

- Incorporates a *convention over configuration* mentality.
- Use the default `*.tpl` extension to find and render templates, or specify your own.
- Use the template extension on a directory to render all files under it.
- Specify the path(s) or let it default to searching for templates in the current working directory.
- Use template variables in filenames to render scalar filename outputs.
- Use hash/dict or list/array types in filenames to render multiple files.
- Default configuration expects a file named `shadowconf` with any of the following extensions: `.json`, `.hcl`, `.env`, `.yml`, `.ini`.
- If no configuration file is specified, it will load and use the shell environment to render variables.
- All defaults can be overriden.

1.2 Quick Install

Install from PyPi:

```
pip install shadowgen
```

Install from GitHub:

```
git clone https://github.com/karma0/shadow
cd shadow
pip install -U .
```

1.3 Examples

Display the help and exit:

```
shadow --help
```

Discover templates to be generated:

```
shadow sim
```

Find all templates in the current working directory and generate them using the config file `shadowconf.json` as the variables to build them:

```
shadow fax
```

Find all generated templates and remove them:

```
shadow clean
```

Generate templates in the `tests` directory on files ending in `*.j2`, using environment variables to fill and render the templates:

```
shadow fax -e -t .j2 tests
```

Generate the single template file named `test.txt` using the HCL config file `test.txt.hcl`:

```
shadow fax -c test.txt.hcl test.txt.tpl
```

1.4 Credits

Created and maintained by karma0.

This package was created with Cookiecutter and the karma0/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Shadow, run this command in your terminal:

```
$ pip install shadowgen
```

This is the preferred method to install Shadow, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Shadow can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/karma0/shadow
```

Or download the tarball:

```
$ curl  -OL https://github.com/karma0/shadow/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Shadow in a project:

```
from shadow.shadow import Shadow
shadow = Shadow()
for tmpl in shadow.run():
    print(f"Generating template--source: {tmpl.source}; destination: {tmpl.
↪destination}")
shadow.render()
```

To use Shadow on the command line, consult the output of the following command:

```
shadow --help
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/karma0/shadow/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Shadow could always use more documentation, whether as part of the official Shadow docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/karma0/shadow/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *shadow* for local development.

1. Fork the *shadow* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/shadow.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv shadow
$ cd shadow/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 shadow tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/karma0/shadow/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_shadow
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Bobby <karma0@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.3.2 (2019-04-01)

- Updated installation documentation.

6.2 0.3.1 (2019-04-01)

- Renamed PyPi project to shadowgen.
- Dependency upgrades.

6.3 0.3.0 (2019-02-06)

- Added config passthrough from CLI.
- Fixed yield bug in rendering of filenames.
- Added some preliminary tests.
- Fixed logging.
- Added always fallback to loading the environment if no config file is present.
- Added checks for shadowconf file using extensions: json, ini, env, etc.

6.4 0.2.2 (2019-01-31)

- Documentation fixes.

6.5 0.2.1 (2019-01-31)

- Version bump; getting everything working.

6.6 0.2.0 (2019-01-31)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search